

Sound and Automated Synthesis of Digital Stabilizing Controllers for Continuous Plants

Alessandro Abate¹, Iury Bessa², Dario Cattaruzza¹, Lucas Cordeiro^{1,2},
Cristina David¹, Pascal Kessel¹ and Daniel Kroening¹

¹University of Oxford, Oxford, United Kingdom

²Federal University of Amazonas, Manaus, Brazil

ABSTRACT

Modern control is implemented with digital microcontrollers, which are designed to work in an embedded environment, within a dynamical plant that represents physical components. We present a new algorithm based on counterexample guided inductive synthesis that automates the design of digital controllers that are correct by construction. The synthesis result is sound with respect to the complete range of approximations, including time discretization, quantization effects, and finite-precision arithmetic and their related rounding errors. We have implemented our new algorithm in a tool called DSSynth, and are able to automatically generate stable controllers for a set of intricate plant models taken from the literature within minutes.

Keywords

Control synthesis, CEGIS, finite word length, time sampling, quantization, interval analysis

1. INTRODUCTION

Modern implementations of embedded control systems have proliferated with the availability of low-cost devices that can perform highly non-trivial control tasks, with significant impact in numerous application areas such as environmental control and robotics [2, 15].

Correct and sound control is non-trivial, however. The problem is exacerbated by artefacts specific to digital control, such as the effects of finite-precision arithmetic, time discretization, and quantization noise introduced by A/D and D/A conversion. Thus, the programming is a key barrier to broad adoption of digital control, and requires considerable expertise.

Beyond classical a-posteriori validation procedures in digital control, there has been plenty of previous work aiming at *verifying* a given controller designed by the engineer. Recent work can show the stability of digital controllers with consideration of implementation aspects, i.e., fixed-point arithmetic and the word length used [6]. The authors of [6] exploit ad-

vances in bit-accurate verification of C programs to obtain a verifier for software-implemented digital control.

By contrast, we leverage a very recent step-change in the scalability of *program synthesis*. Program synthesis engines use a program specification as the starting point, and subsequently generate a sequence of candidate programs. The candidate programs are iteratively refined to match the specification more closely. Program synthesizers that implement Counter-Example Guided Inductive Synthesis (CEGIS) [34] are now able to generate programs for highly non-trivial specifications with a very high degree of automation. Modern CEGIS engines combine automated testing, genetic algorithms and SMT-based automated reasoning [10, 33].

By applying a state-of-the-art CEGIS engine we are now able to present a tool that automatically *generates* digital controllers for a given continuous plant model that are correct by construction. This approach promises to reduce the cost and time of development of digital control dramatically, and requires considerably less expertise than methodologies driven by verification. Specifically, we synthesize controllers for hybrid closed-loop systems, i.e., software-implemented embedded controllers along with a model of their physical environment (the plant), that are *stable*. Due to the complexity of such systems, in this work we focus on linear models with known configurations and perform parametric synthesis of stabilizing digital controllers.

Our work addresses particularly challenging aspects of the control synthesis problem. We perform digital control synthesis over a hybrid model, where the plant exhibits continuous behavior whereas the controller operates in discrete time and over a quantized domain. Inspired by a classical approach [2], we translate the problem into a single digital domain. Given our primary interest in stability despite digitization effects, we have modeled a digital equivalent of the plant by evaluating the effects of the quantizers (A/D and D/A converters) as time discretization elements with quantization noise. The resulting closed-loop system is a program loop that operates on bit-vectors encoded using fixed-point arithmetic with finite word length (FWL). The deterministic effects of the FWL and quantization errors are incorporated into the model as an additive uncertainty, which is taken into account during the CEGIS-based synthesis of the control software for the plant.

In summary, this paper makes the following original contributions.

- We design and implement an automatic approach in our tool DSSynth for generating *correct-by-construction* digital controllers that is based on state-of-the-art inductive synthesis techniques. Our application of pro-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

gram synthesis is non-trivial and requires addressing challenges specific to embedded control systems, such as the effects of quantizers and FWL. In particular, we have found that a two-stage verification engine that continuously refines the precision of the fixed-point representation of the plant is key to the performance of our approach yielding, in most cases, a speed-up of at least two orders of magnitude over a conventional one-stage verification engine.

- We provide experimental results showing that **DSSynth** is able to efficiently synthesize stable controllers for a set of intricate benchmarks taken from the literature: the median runtime for our benchmark set is 197s, i.e., half of the controllers can be synthesized in less than five minutes.

We cover preliminaries on plant discretization, the verification of closed-loop control systems, and stability in Section 2. Our core contribution is in Section 3: we present our CEGIS-based algorithm for generating stable controllers for a given plant model. Our experimental setup and experimental results are given in Section 4, and we discuss how we extend the state-of-the-art beyond the existing literature in Section 5. We conclude and describe future work in Section 6.

2. PRELIMINARIES

2.1 Discretizing the Plant

The digital controllers synthesized using the algorithm we present in this paper are typically used in closed loops with continuous (physical) plants. Thus, the proposed approach has to consider dynamics of continuous parts of the system (i.e., the plant) and of discrete parts (i.e., the digital controller). To obtain a joint model, we fully discretize the continuous plant. We briefly recall the standard foundations of this step.

We only consider transfer function models, and require a z -domain transfer function $G(z)$ that captures all aspects of the continuous plant (with a Laplace domain transfer function $G(s)$) and the sampling and hold process. The continuous model of the plant must be discretized to obtain the corresponding coefficients of $G(z)$.

Among the discretization methods available in the literature [15], the zero-order hold (ZOH) discretization is considered suitable for representing the sample and hold processes that are typically employed in complex systems [18]. The ZOH discretization models the exact effect of sampling and DAC interpolation over the plant.

ASSUMPTION 1. *The sample and hold effects of the ADC and the ZOH of the DAC are synchronized, i.e., there is no delay between sampling the plant output at ADC and updating the DAC accordingly. The DAC interpolator is an ideal ZOH.*

LEMMA 1. [2] *Given a synchronized ZOH input and sample and hold output on the plant with a sample time T satisfying the Nyquist criterion, the discrete pulse transfer function $G(z, T)$ is an exact z -domain representation of $G(s)$ that can be computed using the following formula:*

$$G(z, T) = (1 - z^{-1}) \mathcal{Z} \left\{ \mathcal{L}^{-1} \left\{ \frac{G(s)}{s} \right\} \right\}_{t=kT}. \quad (1)$$

For the sake of brevity, we will use the notation $G(z)$ to represent the pulse transfer function.

Lemma 1 ensures that the poles and zeros match under the $\mathcal{Z} \{ \mathcal{L}^{-1} \{ \cdot \}_{t=kT} \}$ operations, and it includes the ZOH dynamics in the $(1 - z^{-1})$ term, which is sufficient for stability studies over $G(s)$.

2.2 Verifying Closed-loop Control Systems

CEGIS-based control synthesis requires a formal verifier to check whether a candidate controller meets the requirements when combined with the plant. We use the Digital-System Verifier (DSVerifier) [17] for this purpose. It checks the stability of closed-loop control systems and considers finite-word length (FWL) effects in the digital controller and uncertainty parameters in the plant model (plant intervals) [6]. Let $C(z)$ be a digital controller and $G(z)$ be a plant model given as

$$C(z) = \frac{C_n(z)}{C_d(z)} = \frac{\beta_0 + \beta_1 z^{-1} + \dots + \beta_{M_C} z^{-M_C}}{\alpha_0 + \alpha_1 z^{-1} + \dots + \alpha_{N_C} z^{-N_C}}, \quad (2)$$

$$G(z) = \frac{G_n(z)}{G_d(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_{M_G} z^{-M_G}}{a_0 + a_1 z^{-1} + \dots + a_{N_G} z^{-N_G}}. \quad (3)$$

where $\vec{\beta}$ and $\vec{\alpha}$ are vectors containing the controller's coefficients; similarly, \vec{b} and \vec{a} denote the plant's coefficients; and $N_{(\cdot)}$ and $M_{(\cdot)}$ indicate the order of the polynomial of the corresponding block.

Uncertainties in $G(z)$ may appear due to uncertainties in $G(s)$ (i.e., $\Delta_p G_n(s), \Delta_p G_d(s)$, which arise from tolerances/imprecision in the original model), errors in the numeric calculations, or roundoffs in the quantization. These uncertainties are parametrically expressed by additive terms, resulting in an uncertain model $\hat{G}(z)$, such that:

$$\hat{G}(z) = \frac{G_n(z) + \Delta G_n(z)}{G_d(z) + \Delta G_d(z)}, \quad (4)$$

which will be represented by the following transfer function:

$$\hat{G}(z) = \frac{\hat{b}_0 + \hat{b}_1 z^{-1} + \dots + \hat{b}_{M_G} z^{-M_G}}{\hat{a}_0 + \hat{a}_1 z^{-1} + \dots + \hat{a}_{N_G} z^{-N_G}} : N_G \geq M_G. \quad (5)$$

Due to the nature of the methods we use for the stability check, we require that the parametric errors in the plant have the same polynomial order as the plant itself (all other errors described in this paper are ensured to fulfill this property in our implementation).

In order to simplify the presentation, we employ a notation based on the coefficients of the polynomials. Let \mathcal{P}^N be a polynomial domain of order N . Let $P \in \mathcal{P}^{M,N}$ be a rational polynomial $\frac{P_n}{P_d}$, where $P_n \in \mathcal{P}^M$ and $P_d \in \mathcal{P}^N$. For a vector of coefficients

$$\vec{P} \in \mathbb{R}^{N+M+2} = [n_0 \ n_1 \ \dots \ n_M \ d_0 \ d_1 \ \dots \ d_N]^T \quad (6)$$

and an uncertainty vector

$$\Delta \vec{P} = [\Delta n_0 \ \Delta n_1 \ \dots \ \Delta n_M \ \Delta d_0 \ \Delta d_1 \ \dots \ \Delta d_N]^T \quad (7)$$

we write $\vec{\hat{G}} = \vec{G} + \Delta \vec{G}$. In the following we will either use the functions $G(z)$, $C(z)$ directly, or work over their respective coefficients \vec{G} , \vec{C} in vector form.

A typical digital control system with a continuous plant and a discrete controller is illustrated in Figure 1. The DAC and ADC converters introduce quantization errors which are modeled as noises $\nu_1(z)$ and $\nu_2(z)$, $G(s)$ is the continuous-time plant model with parametric additive uncertainty $\Delta_p G_n(s)$ and $\Delta_p G_d(s)$, $R(z)$ is the reference signal, $U(z)$ is the control signal, and $\hat{Y}(z)$ is the output signal

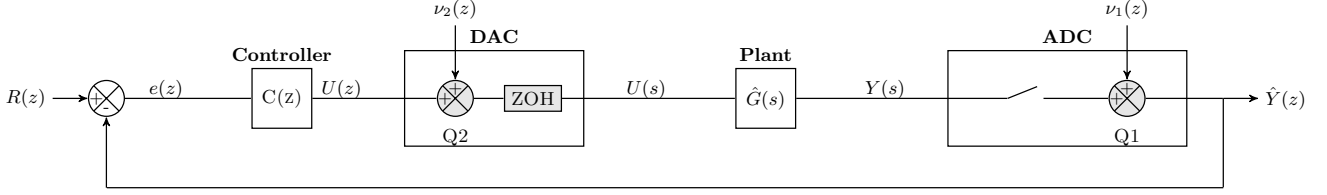


Figure 1: Closed-loop digital control system

affected by the noises and uncertainties in the closed-loop system. The ADC and DAC may be abstracted by transforming the closed-loop system in Figure 1 into the digital system in Figure 2, where the parametric uncertainties are represented by $\Delta_p \vec{G}$ and the entire effect of ν_1 and ν_2 in the output $\hat{Y}(z)$ is represented as an additive output noise $\nu(z)$.

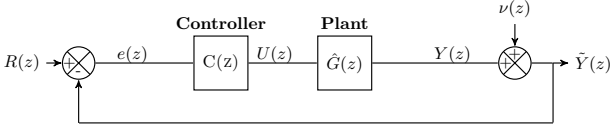


Figure 2: System equivalent to the system in Figure 1

In Figure 2, two sources of uncertainty are illustrated: parametric uncertainties related to modeling errors, and uncertainties introduced by quantizations in sampling and holding process, i.e., ADC and DAC conversions. Assuming linearity and independence between these sources, the uncertain model may be rewritten as a vector of coefficients in the z -domain using equations (6) and (7)

$$\vec{G} = \vec{G} + \Delta_p \vec{G} + \Delta_q \vec{G}, \quad (8)$$

where $\Delta_p \vec{G}$ is the term that corresponds to the parametric uncertainties and $\Delta_q \vec{G}$ is the term that reflects the quantization effects on the plant model.

Since the controller is implemented using a FWL, $C(z)$ also suffers disturbances from the FWL effects, with roundoffs in coefficients that may change the closed-loop poles and zeros position, and consequently affect its stability, as argued in [6].

Let $\hat{C}(z)$ be the transfer function of the digital controller represented using this FWL with integer size I and fractional size F . We shall denote the FWL domain of coefficients as $\mathbb{R}\langle I, F \rangle$ and assume that there exists a function $FWL_n[\cdot] : \mathcal{P}^n \rightarrow \mathcal{P}^n\langle I, F \rangle$, which applies the FWL effects to a polynomial, with an equivalent $FWL_{n,m}[\cdot] : \mathcal{P}^{n,m} \rightarrow \mathcal{P}^{n,m}\langle I, F \rangle$ that applies the same effect to a ratio of polynomials, where \mathcal{P}^n is the space of polynomials of n -th order, $\mathcal{P}^n\langle I, F \rangle$ is the space of polynomials with coefficients in $\mathbb{R}\langle I, F \rangle$, and $\mathcal{P}^{n,m}$, $\mathcal{P}^{n,m}\langle I, F \rangle$ are their rational equivalent.

Thus, the perturbed controller model $\hat{C}(z)$ may be obtained from the original model $C(z) = \frac{C_n(z)}{C_d(z)}$ as follows:

$$\hat{C}(z) = FWL_{M_C, N_C}[C(z)] = \frac{FWL_{M_C}[C_n(z)]}{FWL_{N_C}[C_d(z)]}. \quad (9)$$

In the case of a synthesized controller, because the synthesis is performed using FWL, $\hat{C}(z) \equiv C(z)$.

2.3 Stability under FWL and Noise

Correct synthesis of the digital controller requires consideration of the effect of FWL on the controller and quantization

noises in the closed loop system. Let the quantizer $Q1$ be the source of a white noise ν_1 and $Q2$ be the source of a white noise ν_2 . The following equation models the system in Figure 1, encompassing the parametric uncertainties $\Delta \vec{G}$ and the FWL effects on the controller ($\hat{C}(z)$):

$$\hat{Y}(z) = \hat{C}(z)\hat{G}(z)[R(z) + \nu_1(z)] + \nu_2(z)\hat{G}(z) - \hat{Y}(z)\hat{C}(z)\hat{G}(z), \quad (10)$$

The above can be rewritten as follows:

$$\hat{Y}(z) = H_1[R(z) + \nu_1(z)] + H_2(z)\nu_2(z), \quad (11)$$

where

$$H_1 = \frac{\hat{C}(z)\hat{G}(z)}{1 + \hat{C}(z)\hat{G}(z)}, \quad (12)$$

and

$$H_2 = \frac{\hat{G}(z)}{1 + \hat{C}(z)\hat{G}(z)}. \quad (13)$$

ASSUMPTION 2. The quantization noises ν_1 (from $Q1$) and ν_2 (from $Q2$) are uncorrelated white noises and their amplitudes are always bounded by the half of quantization step [2], i.e., $|\nu_1| \leq \frac{q_1}{2}$ and $|\nu_2| \leq \frac{q_2}{2}$, where q_1 and q_2 , respectively, are the quantization steps of ADC and DAC.

A discrete-time linear system is said to be Bounded-Input and Bounded-Output (BIBO) stable if and only if every pole of its transfer function lies inside the unit circle [3]. Analyzing Eq. (11), the following proposition gives necessary conditions for BIBO stability of the closed-loop system in Figure 1, since the exogenous signals $R(z)$, ν_1 , and ν_2 are bounded.

PROPOSITION 1. [13, 6] Consider a feedback closed-loop control system as given in Figure 1 with a FWL implementation of the digital controller $\hat{C}(z) = FWL_{M_C, N_C}[C(z)]$ and uncertain discrete model of the plant

$$\hat{G}(z) = \frac{\hat{G}_n(z)}{\hat{G}_d(z)} : \vec{G} = \vec{G} + \Delta_p \vec{G} + \Delta_q \vec{G} \quad (\text{from 6,7})$$

Then $\hat{G}(z)$ is BIBO-stable if and only if:

- the roots of characteristic polynomial $S(z)$ are inside the open unit circle, where $S(z)$ is:
$$S(z) = FWL_{M_C}[C_n(z)]\hat{G}_n(z) + FWL_{N_C}[C_d(z)]\hat{G}_d(z); \quad (14)$$
- the direct loop product $FWL_{M_C, N_C}[C(z)] \cdot G(z)$ has no pole-zero cancellation on or outside the unit circle.

Proposition 1 provides the necessary conditions for the digital controller to stabilize the closed-loop system, considering plant parametric uncertainties (i.e., $\Delta_p \vec{G}$), quantization

noises (ν_1 and ν_2) and FWL effects in the control software. Notice that the model for plant uncertainty due to quantization noise (i.e., $\Delta_q \tilde{G}$) may be disregarded if the conditions on Proposition 1 are satisfied and if the quantization noise is bounded. The boundedness of the quantization noise comes from Assumption 2.

3. AUTOMATED PROGRAM SYNTHESIS FOR DIGITAL CONTROL

3.1 Overview of the Synthesis Process

In order to synthesize closed-loop digital control systems, we use a program synthesis engine. Our program synthesizer implements Counter-Example Guided Inductive Synthesis (CEGIS) [34]. We start by presenting its general architecture followed by describing the parts specific to closed-loop control systems. A high-level view of the synthesis process is given in Figure 3. Steps 1 to 3 are performed by the user and Steps A to D are automatically performed by our tool for Digital Systems Synthesis, named **DSSynth**.

Given a model for a plant in ANSI-C syntax as input (Steps 1–3), **DSSynth** constructs a non-deterministic model to represent the plant family (Step A), and formulates a function (Step B) using implementation details provided in Steps 2 and 3 to calculate the parameters of the controller to be synthesized (Step C). Finally, **DSSynth** builds an intermediate ANSI-C code for the digital system implementation, which is used as input for the CEGIS engine (Step D).

This intermediate ANSI-C code model contains a specification ϕ for the property of interest (i.e., robust stability) and is passed to the Counterexample-Guided Inductive Synthesis (CEGIS) module of CBMC [8], where the controller is marked as the input variable to synthesize. CEGIS employs an iterative, counterexample-guided refinement process, which is explained in detail in Section 3.2. CEGIS reports a successful synthesis result if it generates a controller that is safe with respect to ϕ . In particular, the ANSI-C code model guarantees that a synthesized solution is complete and sound with respect to the stability property ϕ , since it does not depend on system inputs and outputs. In the case of stability, the specification ϕ consists of a number of assumptions on the polynomial coefficients, following Jury's Criteria, as well as the restrictions on the representation of these coefficients as discussed in detail in Section 3.3.

3.2 Architecture of the program synthesizer

The input specification provided to the program synthesizer is of the form $\exists \vec{P}. \forall \vec{x}. \sigma(\vec{x}, \vec{P})$ where \vec{P} ranges over functions, \vec{x} ranges over ground terms and σ is a quantifier-free formula. We interpret the ground terms over some finite domain \mathcal{D} .

The design of our synthesizer is given in Figure 4 and consists of two phases, **SYNTHESIZE** and **VERIFY**, which interact via a finite set of test vectors **INPUTS** that is updated incrementally. Given the aforementioned specification σ , the **SYNTH** procedure tries to find an existential witness \vec{P} satisfying the specification $\sigma(\vec{x}, \vec{P})$ for all \vec{x} in **INPUTS** (as opposed to all $\vec{x} \in \mathcal{D}$). If **SYNTHESIZE** succeeds in finding a witness \vec{P} , this witness is a candidate solution to the full synthesis formula. We pass this candidate solution to **VERIFY**, which checks whether it is a full solution (i.e., \vec{P} satisfies the specification $\sigma(\vec{x}, \vec{P})$ for all $\vec{x} \in \mathcal{D}$). If this is the case, then the algorithm terminates. Otherwise, additional information is provided to the **SYNTHESIZE** phase in the form of a new

counterexample that is added to the **INPUTS** set and the loop iterates again (note that, for now, we ignore the second feedback signal “Increase Precision” provided by the **VERIFY** phase in Figure 4 as it is specific to control synthesis and will be described in the next section).

It is worth noting that each iteration of the loop adds a new input to the finite set **INPUTS** that is used for synthesis. Given that the full set of inputs \mathcal{D} is finite, this means that the refinement loop can only iterate a finite number of times.

3.3 Synthesis for control

Specification of the stability property.

Next, we describe the specific property that we pass to the program synthesizer as the specification σ . There are two different algorithms in DSVerifier that can be used for stability verification [5, 6], one based on Schur's decomposition and another one based on Jury's criteria [2]. Here, we choose Jury's method [2], due to its efficiency, and we use this method to check the stability in the z -domain for the characteristic polynomial $S(z)$ defined in (14). We consider the following form for $S(z)$:

$$S(z) = a_0 z^N + a_1 z^{N-1} + \dots + a_{N-1} z + a_N = 0, a_0 \neq 0$$

Next, the following Jury matrix $M = [m_{ij}]_{(2N+2) \times (N+1)}$ is built from $S(z)$ coefficients:

$$M = \begin{pmatrix} V^{(0)} \\ V^{(1)} \\ \vdots \\ V^{(N-2)} \end{pmatrix},$$

where $V^{(k)} = [v_{ij}^{(k)}]_{2 \times N}$ such that:

$$v_{ij}^{(0)} = \begin{cases} a_{j-1}, & \text{if } i = 1 \\ v_{(1)(N-j+1)}^0, & \text{if } i = 2 \end{cases}$$

$$v_{ij}^{(k)} = \begin{cases} 0, & \text{if } j > n - k \\ v_{1j}^{(k-1)} - v_{2j}^{(k-1)} \cdot \frac{v_{11}^{(k-1)}}{v_{21}^{(k-1)}}, & \text{if } j \leq n - k \text{ and } i = 1 \\ v_{(1)(N-j+1)}^{(k-1)}, & \text{if } j \leq n - k \text{ and } i = 2 \end{cases}$$

where $k \in \mathbb{Z}$, such that $0 < k < N - 2$. $S(z)$ is the characteristic polynomial of a stable system if and only if the following four conditions hold: $R_1 : S(1) > 0$; $R_2 : (1)^N S(1) > 0$; $R_3 : |a_0| < a_N$; $R_4 : m_{11} > 0 \wedge m_{31} > 0 \wedge m_{51} > 0 \wedge \dots \wedge m_{(2N-3)(1)} > 0$.

The stability property is then encoded by creating a constraint of the form: $\phi_{\text{stability}} \equiv (R_1 \wedge R_2 \wedge R_3 \wedge R_4)$.

Fixed-point computation in program synthesis.

The program synthesis engine uses fixed-point arithmetic. Specifically, we use the domain $\mathbb{R}\langle I, F \rangle$ for the controller's coefficients and the domain $\mathbb{R}\langle I_p, F_p \rangle$ for the plant's coefficients, where I and F , as well as I_p and F_p , denote the number of bits for the integer and fractional parts, respectively, and $\mathbb{R}\langle I_p, F_p \rangle \supseteq \mathbb{R}\langle I, F \rangle$.

Given the use of fixed-point arithmetic, we need to examine the effect of discretization during these operations. Let $\tilde{C}(z)$ and $\tilde{G}(z)$ be the transfer functions represented using fixed-

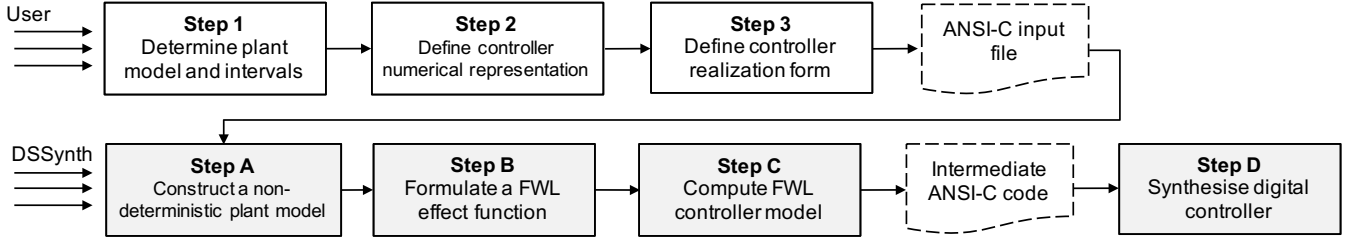


Figure 3: Overview of the synthesis process

point bit vectors.

$$\tilde{C}(z) = \frac{\tilde{\beta}_0 + \tilde{\beta}_1 z^{-1} + \dots + \tilde{\beta}_{M_C} z^{-M_C}}{\tilde{\alpha}_0 + \tilde{\alpha}_1 z^{-1} + \dots + \tilde{\alpha}_{N_C} z^{-N_C}}, \quad (15)$$

$$\tilde{G}(z) = \frac{\tilde{b}_0 + \tilde{b}_1 z^{-1} + \dots + \tilde{b}_{M_G} z^{-M_G}}{\tilde{a}_0 + \tilde{a}_1 z^{-1} + \dots + \tilde{a}_{N_G} z^{-N_G}}. \quad (16)$$

Since the controller is synthesized in the $\mathbb{R}\langle I, F \rangle$ domain,

$$\forall i \leq N_C, j \leq M_C \quad \tilde{\beta}_i \equiv \beta_i \wedge \tilde{\alpha}_j \equiv \alpha_j \Leftrightarrow \tilde{C}(z) \equiv C(z)$$

However, given a real plant $\hat{G}(z)$, and a discretizing function

$$f^p : \mathbb{R} \rightarrow \mathbb{R}\langle I_p, F_p \rangle \triangleq \tilde{x} = f^p(x) : \tilde{x} = x - (x \bmod \tilde{x}) \quad (17)$$

$$FWL_n^p[P \in \mathcal{P}] = \tilde{P} \in \mathcal{P}^n\langle I_p, F_p \rangle : c_i \in \tilde{P} \wedge \tilde{c}_i \in \tilde{P} = f^p(c_i)$$

we calculate $\tilde{G}(z) = FWL_{M_G, N_G}^p(\hat{G}(z))$

$$\begin{aligned} \tilde{G}(z) &= \frac{(\hat{b}_0 + \Delta_b \hat{b}_0) + \dots + (\hat{b}_{M_G} + \Delta_b \hat{b}_{M_G}) z^{-M_G}}{(\hat{a}_0 + \Delta_b \hat{a}_0) + \dots + (\hat{a}_{N_G} + \Delta_b \hat{a}_{N_G}) z^{-N_G}} \\ &\Rightarrow \tilde{\vec{G}} = \vec{\tilde{G}} + \Delta_b \vec{\tilde{G}} = \vec{\tilde{G}} + \Delta_p \vec{\tilde{G}} + \Delta_b \vec{\tilde{G}}. \end{aligned} \quad (18)$$

where $\Delta_b \hat{c}_i = \hat{c}_i \bmod \tilde{c}_i$ and $\Delta_b G$ represents the plant uncertainty caused by the rounding off effect. We may encompass all the uncertainty as $\Delta \vec{\tilde{G}} = \Delta_p \vec{\tilde{G}} + \Delta_b \vec{\tilde{G}}$. Since the complexity of SAT solvers is worst-case exponential in the size of the problem instance, our algorithm tries to solve the problem first for a low I_p, F_p , iteratively increasing the precision if it is insufficient.

Our synthesis problem.

The synthesis problem we are trying to solve is the following: find a digital controller $C(z)$ that makes the closed-loop system stable for all possible uncertainties $\tilde{G}(z)$ (18). When mapping back to the notation used for describing the general architecture of the program synthesizer, the controller $C(z)$ denotes P and $\tilde{G}(z)$ represents x .

As mentioned above, we compute the coefficients for $C(z)$ in the domain $\mathbb{R}\langle I, F \rangle$, and those for $\tilde{G}(z)$ in the domain $\mathbb{R}\langle I_p, F_p \rangle$. While the controller's precision $\langle I, F \rangle$ is given, we can vary $\langle I_p, F_p \rangle$ such that $\mathbb{R}\langle I_p, F_p \rangle \supseteq \mathbb{R}\langle I, F \rangle$.

3.4 The SYNTHESIZE and VERIFY phases

The SYNTHESIZE phase uses BMC to compute a solution $C(z)$.

There are two approaches for the VERIFY phase. The first approach uses interval arithmetic [26] to represent the coefficients $[c_i - \Delta_p c_i - \Delta_b c_i, c_i + \Delta_p c_i + \Delta_b c_i + (2^{-F_p})]$ and rounds operations outwards during calculations. This effectively allows us to simultaneously evaluate the whole collection of plants \mathcal{G} plus the effects of numeric calculations

in the engine. Synthesized controllers using this verifier will always be stable for all plants in the family. Our preliminary experiments showed that a synthesis approach using this verification engine has poor performance and we therefore designed a second approach, which we describe below. Our experimental results described in Section 4 show that the speedup yielded by the second approach is in most cases of at least two orders of magnitude.

The second approach is illustrated in Figure 4 and uses a two-stages verification: the first stage performs potentially unsound fixed-point operations assuming a plant precision $\langle I_p, F_p \rangle$, and the second stage restores soundness by validating these operations using interval arithmetic on the synthesized controller. In more detail, in the first stage, denoted by UNCERTAINTY in Figure 4, assuming a precision $\langle I_p, F_p \rangle$ we check whether the system is unstable for the current candidate solution, i.e., if $\neg \phi_{\text{stability}}$ is satisfiable for $S(z)$. If this is the case, then we obtain a counterexample $\tilde{G}(z)$, which makes the closed-loop system unstable. This uncertainty is added to the set INPUTS such that, in the subsequent SYNTHESIZE phase, we obtain a candidate solution consisting of a controller $C(z)$, which makes the closed-loop system stable for all the uncertainties accumulated in INPUTS.

If the UNCERTAINTY verification stage concludes that the system is stable for the current candidate solution, then we pass this solution to the second verification stage, PRECISION, which checks the propagation of the error in the fixed-point calculations using a Fixed-point Arithmetic Verifier based on interval arithmetic.

If the PRECISION verification returns *false*, then we increase the precision of $\langle I_p, F_p \rangle$ and re-start the SYNTHESIZE phase with an empty INPUTS set. Otherwise, we found a full sound solution for our synthesis problem and we are done.

In the rest of the paper, we will refer to the two approaches for the VERIFY phase as one-stage and two-stage, respectively.

3.5 Illustrative Example

We illustrate our approach with a classical cruise control example, extracted from the literature [3]. The example highlights the challenges that arise when using finite-precision arithmetic in digital control. We are given a discrete plant model (with a time step of 0.2s), which is represented by the following z -expression:

$$G(z) = \frac{0.0264}{z - 0.9998}. \quad (19)$$

Using an optimization tool, the authors of [35] have designed a high-performance controller for this plant, which is characterized by the following z -domain transfer function:

$$C(z) = \frac{2.72z^2 - 4.153z + 1.896}{z^2 - 1.844z + 0.8496}. \quad (20)$$

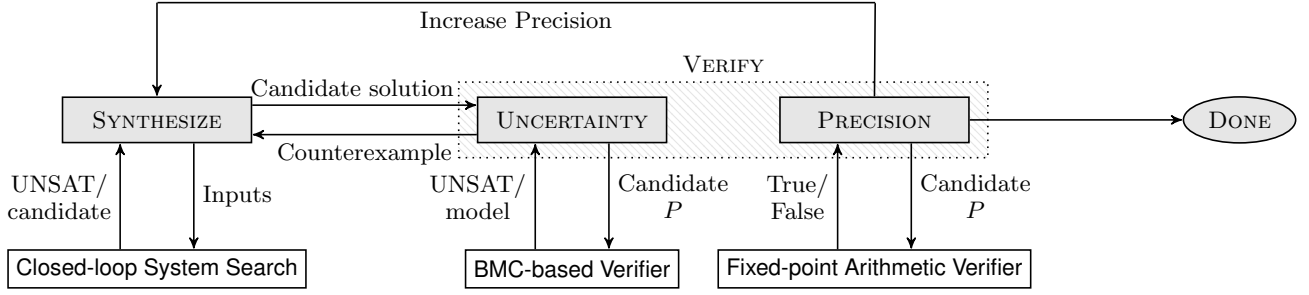


Figure 4: Counterexample-Guided Inductive Synthesis of Closed-loop Systems (Step D)

The authors of [35] claim that this controller $C(z)$ (Eq. (20)) stabilizes the closed-loop system for the discrete plant model $G(z)$ (Eq. 19). However, if the effects of finite-precision arithmetic are considered during verification (e.g., fixed-point arithmetic and word-length), then this closed-loop system becomes unstable. For instance, an implementation of $C(z)$ using $\langle 4, 16 \rangle$ fixed-point numbers (i.e., 4 bits for the integer part and 16 bits for the fractional part) can be modeled as follows:

$$C_q(z) := \frac{2.7199859619140625z^2 - 4.1529998779296875z + 1.89599609375}{z^2 - 1.843994140625z + 0.8495941162109375} \quad (21)$$

The resulting system using the typical series loop configuration, where $C_q(z)$ and $G(z)$ are in the forward path, is unstable. Figure 5a gives the Bode diagram for the digital controller represented by Eq. (20). The phase margin is negative, which means that the controller represented by Eq. (20) is unstable, considering FWL effects.

3.6 Program synthesis for the example

We now demonstrate how our approach solves the synthesis problem for the example given in the previous section. We start with the candidate solution with all coefficients zero, as given below, and a precision of $I_p = 16$, $F_p = 24$. The digital controller is designed with standard techniques for control systems [25, 27].

$$C(z) = \frac{0z^2 + 0z + 0}{0z^2 + 0z + 0}$$

In the first VERIFY stage, the UNCERTAINTY check finds the following counterexample:

$$\tilde{G}(z) = \frac{0.026506}{1.000610z + 1.002838}$$

We add this counterexample to INPUTS and initiate the SYNTHESIZE phase, where we obtain the following candidate solution:

$$C(z) = \frac{12.402664z^2 - 11.439667z + 0.596756}{4.003906z^2 - 0.287949z + 0.015625}$$

This time, the UNCERTAINTY check does not find any counterexample and we pass the current candidate solution to the PRECISION verification stage. We obtain the result *false*, meaning that the current precision is insufficient. Consequently, we increase our precision to $I_p = 20$, $F_p = 28$. Since the previous counterexamples were obtained at lower precision, we remove them from the set of counterexamples. Back in the SYNTHESIZE phase, we re-start the process with a candidate solution with all coefficients 0. Next, the UNCERTAINTY verification stage provides the first counterexample

at higher precision:

$$\tilde{G}(z) = \frac{0.026314}{0.999024z - 1.004785}$$

In the next SYNTHESIZE phase, we get a new candidate solution that eliminates the new, higher precision counterexample:

$$C(z) = \frac{11.035202z^2 + 5.846100z + 4.901855}{1.097901z^2 + 0.063110z + 0.128357}$$

This candidate solution is validated as the final solution by both stages UNCERTAINTY and PRECISION in the VERIFY phase. Figure 5 compares the Bode diagram of the illustrative example using the digital controller represented by Eq. (20) from [35] (Figure 5a) and the final candidate solution from our synthesizer (Figure 5b). The final solution from DSSynth is stable since it presents an infinite phase margin and a gain margin of 17.8 dB.

Figure 6 illustrates the step responses of the closed-loop system with the original controller represented by Eq. (20) (Figure 6a), the first (Figure 6b) and final (Figure 6c) candidate solutions provided by DSSynth. The step response in Figure 6a confirms the stability loss if we consider FWL effects. However, Figure 6b shows that the first candidate controller is able to stabilize the closed-loop system without uncertainties, but it is rejected during the PRECISION phase by DSSynth since this solution is not considered to be sound. Finally, Figure 6c shows a stable behavior for the final (sound) solution, which presents a lower settling time.

4. EXPERIMENTAL EVALUATION

This section is split into three parts. Section 4.1 discusses the experimental setup and our benchmarks, while Section 4.2 describes the experimental objectives. Section 4.3 presents the experimental results obtained on the benchmarks with our tool DSSynth.

4.1 Description of the Benchmarks

The first set of benchmarks uses the discrete plant of a cruise control model for a car, and accounts for rolling friction, aerodynamic drag, and the gravitational disturbance force [3]. The plant model is given as follows:

$$G_1(z) = \frac{0.0264}{z - 0.998}$$

The second set of benchmarks considers a simple spring-mass damper [35], where the discrete plant dynamics is represented by the following z -expression:

$$G_2(z) = \frac{5 \times 10^{-5}z + 5 \times 10^{-5}}{z^2 - 2z + 1.0001}.$$

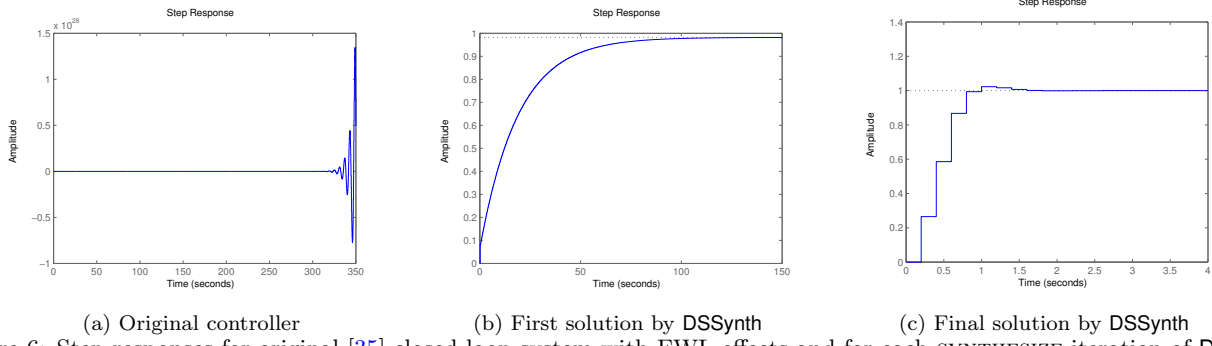


Figure 6: Step responses for original [35] closed-loop system with FWL effects and for each SYNTHESIZE iteration of DSSynth

A third set of benchmarks uses a physical plant for satellite applications [15]. Satellites require attitude (pose) control for orientation of antennas and sensors w.r.t. earth. The satellite attitude control is typically used for three-axis attitude tracking, but here we consider only one axis at a time. The motion equation for the one-axis system, disregarding disturbance torques, is given by

$$I \cdot \ddot{\theta} = \tau_C, \quad (22)$$

where I is the inertia moment of the satellite about its mass center, τ_C is the control torque applied by the thrusters, and θ is the controlled attitude angle. Normalizing Eq. (22) by defining the control signal $u = \frac{\tau_C}{I}$ and taking the Laplace transform, the following s -domain transfer function $G(s)$ is obtained:

$$G_3(s) = \frac{\theta(s)}{u(s)} = \frac{1}{s^2}, \quad (23)$$

Using the ZOH discretization defined in Eq. (1), we get the following z -domain model:

$$G_3(z) = \frac{T^2}{2} \frac{z+1}{(z+1)^2}.$$

The fourth (and last) set of benchmarks considers the following description for the plant $G_4(s)$, which is typically used for evaluating stability margins [21, 22]:

$$G_4(s) = \frac{s-1}{s^2-s-2}, \quad (24)$$

We can obtain $G_4(z)$ using the ZOH discretization defined in Eq. (1) with three different sample times (0.1, 0.05, and 0.03 seconds), respectively, as follows:

$$G_{4a}(z) = \frac{0.100342181002722z - 0.110876810062963}{1.0z^2 - 2.12624017619613z + 1.10517091807565},$$

$$G_{4b}(z) = \frac{0.0500422033454653z - 0.0526068264456340}{1.0z^2 - 2.05640034257636z + 1.05127109637602},$$

$$G_{4c}(z) = \frac{0.0300090687252212z - 0.0309228417953966}{1.0z^2 - 2.03228208009387z + 1.03045453395352}.$$

All experiments were conducted on a 12-core 2.40 GHz Intel Xeon E5-2440 with 96 GB of RAM and Linux OS. All times given are wall clock time in seconds as measured by the UNIX date command. For the approach using a two-stage verification engine we apply a timeout of 8 hours per benchmark, and 24 hours for the approach using a one-stage engine.

4.2 Objectives

Using the closed-loop control system benchmarks given in Section 4.1, our experimental evaluation aims to answer two research questions:

RQ1 (**performance**) does the CEGIS approach generate a FWL digital controller in a reasonable amount of time?

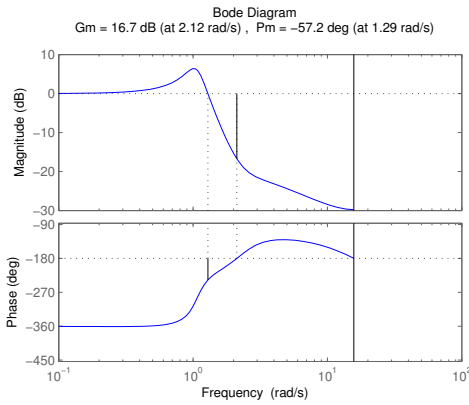
RQ2 (**sanity check**) are the synthesized controllers sound and can their stability be confirmed outside of our model?

4.3 Results

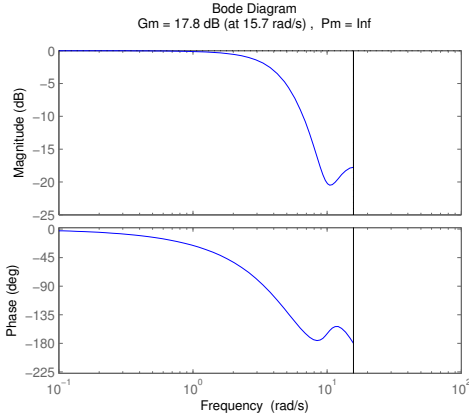
We give the runtimes required to synthesize a stable controller for each benchmark in Table 1. Here, *Plant* is the discrete or continuous plant model, *Benchmark* is the name of the employed benchmark, I and F represent the number of integer and fractional bits of the stable controller, respectively, while *Time* is the total time (in seconds) required to synthesize a stable controller for the given plant.

For the majority of our benchmarks, the conjecture explained in Section 3.3 holds and the two-stage verification engine is able to find a stable solution in less than five minutes. This is possible if the inductive solutions need to be refined with few counterexamples and increases of the fixed-point precision. However, two benchmarks in our set required too many counterexamples to refine their solution (marked with * in Table 1). In these cases, the one-stage engine is able to complement the two-stage approach and synthesize a solution instead, albeit with significantly larger runtimes. This performance difference is due to the fact that the one-stage verification engine does not take advantage of the inductive conjecture inherent to CEGIS, but instead fully explores the counterexample space. The result is a performance difference of at least two orders of magnitude in our experiments, leading to the one-stage engine timing out on the majority of our benchmarks. Table 1 only lists the faster of the two verification engines for each benchmark, which in 12 out of 15 benchmarks is the two-stage one.

The median runtime for our benchmark set is 197s, implying that DSSynth can synthesize half of the controllers in less than 5 minutes. Overall, the average synthesis time amounts to approximately 30 minutes. We consider these times short enough to be of practical use to control engineers, and thus affirm RQ1. We further observe that the approach using the two-stage verification engine is able to synthesize stable controllers in 12 out of the 15 benchmarks, and can be complemented using the one-stage engine for two additional benchmarks, where the inductive conjectures fail. Both verification engines together enable controller synthesis for 14



(a) Original controller [35].



(b) Controller synthesized by DSSynth.

Figure 5: Bode diagram for original [35] and synthesized closed-loop systems.

out of 15 benchmarks. For the one remaining benchmark our approach failed to synthesize a stable controller within the time limits. This can be addressed by either increasing the fixed-point word widths considered or using floating-point arithmetic instead. The synthesized controllers were confirmed to be stable outside of our model representation using MATLAB, positively answering RQ2. A link to the full experimental environment, including scripts to reproduce the results, all benchmarks and the DSSynth tool, is provided in the footnote.¹

4.4 Threats to validity

We report a favourable assessment of DSSynth over a diverse set of real-world benchmarks. Nevertheless, this set of benchmarks is limited within the scope of this paper and DSSynth’s performance may not generalize to other benchmark scenarios. This can be addressed by expanding the benchmark set in future experiments.

Furthermore, our algorithm to select suitable FWL word widths to model plant behavior employs a heuristic based on user-provided controller word-width specifications. Given the encouraging results of our benchmarks, this heuristic appears to be strong enough for the current benchmark set, but this may not generalize. Further experiments towards determining suitable plant FWL configurations may thus be

¹<https://drive.google.com/file/d/0ByIexo3Z5N91QkhrMi1rdkZFTWc/view>

#	Plant	Benchmark	I	F	Time
1	G_1	CruiseControl02	4	16	17 s
2	G_1	CruiseControl02 [†]	4	16	46 s
3	G_2	SpringMassDamper	15	16	28 s
4	G_2	SpringMassDamper [†]	15	16	✗
5	G_3	SatelliteB2	3	7	7 s
6	G_3	SatelliteB2 [†]	3	7	6601 s*
7	G_3	SatelliteC2	3	5	2 s
8	G_3	SatelliteC2 [†]	3	5	76 s*
9	G_{4a}	a_ST1_IMPL1	16	4	2704 s
10	G_{4a}	a_ST1_IMPL2	16	8	538 s
11	G_{4a}	a_ST1_IMPL3	16	12	12 s
12	G_{4b}	a_ST2_IMPL1	16	4	318 s
13	G_{4b}	a_ST2_IMPL2	16	8	967 s
14	G_{4b}	a_ST2_IMPL3	16	12	9798 s
15	G_{4c}	a_ST3_IMPL1	16	4	6304 s

* = solved with the one-stage engine, † = with uncertainty

Table 1: Experimental results obtained with DSSynth

necessary in future work.

Finally, the experimental results obtained using DSSynth for stability properties may not generalize to other properties. The inductive nature of the two-stage back-end of DSSynth increases performance significantly compared to the one-stage back-end, but this performance benefit introduced by CEGIS’ inductive generalizations may not be observed for other controller properties. Additional experiments are necessary to confirm that the performance of our inductive synthesis approach can be leveraged in those scenarios as well.

5. RELATED WORK

Robust synthesis of Linear Systems.

The problem of parametric control synthesis based on stability measures for continuous Linear Time Invariant (LTI) Single Input-Single Output (SISO) systems has been researched for several decades. On a theoretical level it is a solved problem [36], for which researchers continuously seek better results for a number of aspects in addition to stability. A vast range of pole placement techniques such as Moore’s algorithm for eigenstructure assignment [23] or the more recent Linear Quadratic Regulator (LQR) [4] have been used in several studies with increasing degrees of success to ensure stability. The latter approach highlights the importance of conserving energy during the control process, which results in lower running costs. Since real systems are subject to tolerance and noise as well as the need for economy, more recent studies focus on the problem of achieving robust stability with minimum gain [32, 24]. However, when applied with the aim of synthesising a digital controller, many of these techniques lack the ability to produce sound or stable results because they disregard the effects of quantization and rounding. Recent papers on implementations/synthesis of LTI digital controllers [9, 16] focus on time discretization, failing to account for these error-inducing effects and can therefore result in digital systems that are unstable even though they have been proven to be robustly stable in a continuous space.

Formal Verification of Linear Digital Controllers.

Researchers in formal verification of control systems have

studied various effects of discretizing the dynamics, including delayed response [11], and Finite Word Length (FWL) semantics with the goal to either verify (e.g. [5]) or optimize (e.g. [28]) given implementations.

There are two different problems that arise from FWL semantics. The first is the error in the dynamics caused by the inability to represent the exact state of the physical system while the second relates to rounding errors during computation. In [14], a stability measure based on the error of the digital dynamics ensures that the deviation introduced by FWL does not make the digital system unstable. A more recent approach [37] uses μ calculus to directly model the digital controller so that the selected parameters are stable by design. Most work in verification focusses on finding a correct variant of a known controller, e.g., by looking for optimal parameter representations using FWL, but ignore the effects of rounding errors due to issues of mathematical tractability. The analyses in [35, 31] rely on an invariant computation on the discrete system dynamics using Semi-Definite Programming (SDP). While the former uses BIBO properties to determine stability, the latter uses Lyapunov-based quadratic invariants. In both cases, the SDP solver uses floating-point arithmetic and soundness is checked by bounding the error. An alternative approach is taken by [29], where the verification of existing code is performed against a known model by extracting an LTI model of the code through symbolic execution. In order to account for rounding errors, an upper bound is introduced in the verification phase. If the error of the implementation is lower than this tolerance level, then the verification is successful.

Robust Synthesis of FWL Digital Controllers.

There is no technique in the existing literature for automatic synthesis of fixed-point digital controllers that considers FWL effects. Parameter synthesis tools such as [7] use bounded model checking and fixed-point computations to compute parameters based on user-defined specifications, but they often operate in the continuous domain and have no direct means of evaluating robustness.

Other tools such as [12] are directly aimed at robust stability problems, but they fail to take the effects of FWL into account. In order to provide a correct-by-design digital controller, [1] requires a user-defined finite-state abstraction to synthesize a digital controller based on high-level specifications. While this approach overcomes the challenges presented by the FWL problem, it still requires error-prone user intervention. A different solution that uses FWL as the starting point is an approach that synthesizes word lengths for known control problems [20]; however, this provides neither an optimal result nor a comprehensive solution for the problem.

CEGIS.

Program synthesis is the problem of computing correct-by-design programs from high-level specifications, and algorithms for this problem have made substantial progress in recent years. One such approach [19] inductively synthesizes invariants to generate the desired programs.

Program synthesizers are an ideal fit for synthesis of parametric controllers since the semantics of programs capture effects such as FWL precisely. In [30], the authors use CEGIS for the synthesis of switching controllers for stabilizing continuous-time plants with polynomial dynamics. The work extends to its application on affine systems, finding its

major challenge in the hardness of solving linear arithmetic with the state-of-the-art SMT solvers. Since this approach uses switching states instead of linear dynamics in the digital controller, it entirely circumvents the FWL problem. It is also not suitable for the kind of control we seek to synthesize. We require a combination of a synthesis engine with a control verification tool that addresses the challenges presented here in the form of FWL effects and stability measures for LTI SISO controllers. We take the former from [10] and the latter from [5] while enhancing the procedure by evaluating the quantization effects of the Hardware interfaces (ADC/DAC) to obtain an accurate discrete-time FWL representation of the continuous dynamics.

6. CONCLUSIONS

We have presented a CEGIS-based approach implemented in a tool called **DSSynth**, which uses inductive synthesis in conjunction with an algorithm for verifying robust closed-loop stability. In particular, **DSSynth** is able to compute a parametric solution for a stabilizing digital controller given a continuous (or discrete) plant specification. This enables us to automatically synthesize correct-by-construction digital controllers for a given model of their physical environment. Our approach addresses the effects of the quantizers (A/D and D/A converters) as time discretization elements with quantization noise. Our experimental results show that **DSSynth** is able to synthesize stable controllers in most benchmarks within a reasonable amount of time fully automatically.

Future work will be the extension of this CEGIS-based approach to further classes of systems, including state-space representation. We will also consider performance requirements while synthesizing the digital controller.

7. REFERENCES

- [1] R. Alur, S. Moarref, and U. Topcu. Compositional synthesis with parametric reactive controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 215–224. ACM, 2016.
- [2] K. Åström and B. Wittenmark. *Computer-controlled systems: theory and design*. Prentice Hall information and system sciences series. Prentice Hall, 1997.
- [3] K. J. Astrom and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, USA, 2008.
- [4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.
- [5] I. Bessa, H. Ismail, L. Cordeiro, and J. Filho. Verification of fixed-point digital controllers using direct and delta forms realizations. *Design Autom. for Emb. Sys.*, 20(2):95–126, 2016.
- [6] I. Bessa, H. Ismail, R. Palhares, L. Cordeiro, and J. E. C. Filho. Formal non-fragile stability verification of digital control systems with uncertainty. *IEEE Transactions on Computers (to appear)*, 2016.
- [7] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. Parameter synthesis with IC3. In *FMCAD*, pages 165–168, 2013.
- [8] E. M. Clarke, D. Kroening, and F. Lerda. A tool for checking ANSI-C programs. In *10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 2988, pages 168–176. LNCS, 2004.

- [9] S. Das, I. Pan, K. Halder, S. Das, and A. Gupta. LQR based improved discrete PID controller design via optimum selection of weighting matrices using fractional order integral performance index. *Applied Mathematical Modelling*, 37(6):4253–4268, 2013.
- [10] C. David, D. Kroening, and M. Lewis. Using program synthesis for program analysis. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-20)*, LNCS, pages 483–498. Springer, 2015.
- [11] P. S. Duggirala and M. Viswanathan. Analyzing real time linear control systems using software verification. In *IEEE Real-Time Systems Symposium*, pages 216–226, Dec 2015.
- [12] C. Economakos, G. Economakos, M. Skarpetis, and M. Tzamtzi. Automated synthesis of an FPGA-based controller for vehicle lateral control. In *MATEC Web of Conferences*, volume 41, page 02004. EDP Sciences, 2016.
- [13] S. Fadali and A. Visioli. *Digital Control Engineering: Analysis and Design*, volume 303 of *Electronics & Electrical*. Elsevier/Academic Press, 2009.
- [14] I. J. Fialho and T. T. Georgiou. On stability and performance of sampled-data systems subject to wordlength constraint. *IEEE Transactions on Automatic Control*, 39(12):2476–2481, 1994.
- [15] G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Pearson, 7th edition, 2015.
- [16] S. Ghosh, R. K. Barai, S. Bhattacharya, P. Bhattacharyya, S. Rudra, A. Dutta, and R. Pyne. An FPGA based implementation of a flexible digital PID controller for a motion control system. In *Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE, 2013.
- [17] H. Ismail, I. Bessa, L. C. Cordeiro, E. B. de Lima Filho, and J. E. C. Filho. DSVerifier: A bounded model checking tool for digital systems. In *Model Checking Software (SPIN)*, volume 9232, pages 126–131. LNCS, 2015.
- [18] R. Istepanian and J. F. Whidborne. *Digital controller implementation and fragility: A modern perspective*. Springer Science & Business Media, 2012.
- [19] S. Itzhaky, S. Gulwani, N. Immerman, and M. Sagiv. A simple inductive synthesis methodology and its applications. In *ACM Sigplan Notices*, volume 45, pages 36–46. ACM, 2010.
- [20] S. Jha and S. A. Seshia. SWATI: Synthesizing wordlengths automatically using testing and induction. *arXiv preprint arXiv:1302.1920*, 2013.
- [21] L. Keel and S. Bhattacharyya. Robust, fragile, or optimal? *IEEE Trans. on Automatic Control*, 42(8):1098–1105, 1997.
- [22] L. Keel and S. Bhattacharyya. Stability margins and digital implementation of controllers. In *Proc. American Control Conference*, volume 5, pages 2852–2856, 1998.
- [23] G. Klein and B. Moore. Eigenvalue-generalized eigenvector assignment with state feedback. *IEEE Transactions on Automatic Control*, 22(1):140–141, 1977.
- [24] U. Konigorski. Pole placement by parametric output feedback. *Systems & Control Letters*, 61(2):292–297, 2012.
- [25] B. C. Kuo and F. Golnaraghi. *Automatic Control Systems*. John Wiley & Sons, Inc., 8th edition, 2002.
- [26] R. E. Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, 1966.
- [27] K. Ogata. *Discrete-time Control Systems*. Prentice-Hall, Inc., 1987.
- [28] A. K. Oudjida, N. Chaillet, A. Liacha, M. L. Berrandjia, and M. Hamerlain. Design of high-speed and low-power finite-word-length pid controllers. *Control Theory and Technology*, 12(1):68–83, 2014.
- [29] J. Park, M. Pajic, I. Lee, and O. Sokolsky. Scalable verification of linear controller software. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 662–679. Springer, 2016.
- [30] H. Ravanbakhsh and S. Sankaranarayanan. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *54th IEEE Conference on Decision and Control, CDC*, pages 4232–4239, 2015.
- [31] P. Roux, R. Jobredeaux, and P. Garoche. Closed loop analysis of control command software. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC*, pages 108–117, 2015.
- [32] R. Schmid, L. Ntogramatzidis, T. Nguyen, and A. Pandey. A unified method for optimal arbitrary pole placement. *Automatica*, 50(8):2150–2154, 2014.
- [33] R. Sharma and A. Aiken. From invariant checking to invariant inference using randomized search. In *Computer Aided Verification (CAV)*, pages 88–105, 2014.
- [34] A. Solar-Lezama. Program sketching. *STTT*, 15(5-6):475–495, 2013.
- [35] T. E. Wang, P. Garoche, P. Roux, R. Jobredeaux, and E. Feron. Formal analysis of robustness at model and code level. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC*, pages 125–134, 2016.
- [36] W. Wonham. On pole assignment in multi-input controllable linear systems. *IEEE Transactions on Automatic Control*, 12(6):660–665, 1967.
- [37] J. Wu, G. Li, S. Chen, and J. Chu. Robust finite word length controller design. *Automatica*, 45(12):2850–2856, 2009.